第一章：

一. 1. 操作系统是管理硬件设备，提高其利用率和系统吞吐量，为用户和应用程序提供接口的软件。

4. 实时操作系统是一种为满足严格时间约束而设计的操作系统，能在确保任务在确定时间范围内及时完成响应和处理。

5. 互斥共享是指在多进程或多线程环境中，通过互斥机制确保同一时间只有一个进程或线程可访问共享资源，从而避免资源冲突。

三. 1. 硬件系统. 软件系统.

3. 硬件资源. 软件资源

5. 共享性. 异步性.

第二章：

一. 3. 临界区是进程中访问临界资源的代码段

4. 进程同步是指协调多个进程的执行顺序和速度，确保它们对共享资源的访问符合规则，避免竞争和死锁.

三. 1. 动态性. 异步性. 调度性.

5. 动态. 静态.

6. 间接制约

8. 共享内存. 消息传递. 管道

9. 就绪. 运行. 阻塞.

五. 2. S1是S2, S3的前驱; S2是S4, S5的前驱; S3是S6的前驱; S4, S5, S6是S7的前驱.

semaphore sem_S2=0, sem_S3=0, sem_S4=0, sem_S5=0, sem_S6=0, sem_S7=0;

void S1(){ 执行S1; V(sem_S2); V(sem_S3);} void S2(){ P(sem_S2); 执行S2; V(sem_S4); V(sem_S5);} void S3(){ P(sem_S3); 执行S3; V(sem_S6);} void S4(){ P(sem_S4); 执行S4; V(sem_S7);} void S5(){ P(sem_S5); 执行S5; V(sem_S7);} void S6(){ P(sem_S6); 执行S6; V(sem_S7);} void S7(){ P(sem_S7); P(sem_S7); P(sem_S7); 执行S7;} int main(){ S1(); S2(); S3(); S4(); S5(); S6(); S7(); return 0;}

3. semaphore empty=5, orange=0, apple=0, mutex=1;

```
Dad(){                  Son(){                  Daughter(){
  while(1){               while(1){               while(1){
    P(empty);               P(orange);              P(apple);
    P(mutex);               P(mutex);               P(mutex);
    放入水果;               取橘子;                 取苹果;
    V(mutex);               V(mutex);               V(mutex);
    if(橘子)                V(empty);               V(empty);
     V(orange);             吃橘子;                 吃苹果;
    else                    }                       }
     V(apple);            }                       }
  }
}
```

4. semaphore empty1=m; empty2=n; full1=0; full2=0; mutex1=1; mutex2=1;

```
P1(){                   P2(){                   P3(){
  while(1){               while(1){               while(1){
    从磁盘读出一个文件记录;   P(full1);             P(full2);
    P(empty1);              P(mutex1);              P(mutex2);
    P(mutex1);              从缓冲区1读一个记录;     从缓冲池2读出一个记录打印;
    将文件记录读入缓冲池1;    V(mutex1)               V(mutex2);
    V(mutex1);              V(empty1);              V(empty2);
    V(full1);               P(empty2);            }
  }                         P(mutex2);          }
}                           将文件读入缓冲区2;
                            V(mutex2);
                            V(full2);
                          }
                        }

main(){ cobegin P1(); P2(); P3(); coend }
```

5. int count=0; semaphore mutex=A, sofa=N, empty=A, full=0;
semaphore payment=0, receipt=0;

顾客(){ P(mutex);
if(count>=N+A){ V(mutex); 离开理发店; }else{ count=count+A; V(mutex);
if(count>A){ P(sofa); 就座; P(empty); 离开沙发; V(sofa); } else P(empty);
理发椅上就座; V(full); 理发; 付费; V(payment); P(receipt); 离开理发椅; V(empty);
P(mutex); count=count-A; V(mutex); 离店; }

理发师() {
  while(1){
    P(full);
    理发;
    P(payment);
    收费;
    V(receipt);}
  }.

main(){cobegin 顾客(); 理发师(); coend;}

第三章：

一. 2. 处理机调度是操作系统中为配CPU资源给进程的核心机制，包括：FCFS，SJF，RR，优先级调度多级反馈队列；

3. 周转时间指进程从提交到完成所经历的总时间，周转时间=完成时间-提交时间；

4. 死锁是两个或多个进程因其等待资源为另一进程持有资源，而此进程也恰好在等待原进程所保持资源而陷入永久阻塞的状态.

三. 2. 提交，后备，完成.

6. 预防死锁，避免死锁，检测死锁，解除死锁.